StructSum Generation for Faster Text Comprehension

Parag Jain[1*] **, Andreea Marzoca**[2]**, Francesco Piccinno**[2]
School of Informatics, University of Edinburgh[1]
Google DeepMind[2]
parag.jain@ed.ac.uk, {andreeam,piccinno}@google.com

Abstract

We consider the task of generating structured representations of text using large language models (LLMs). We focus on tables and mind maps as representative modalities. Tables are more organized way of representing data, while mind maps provide a visually dynamic and flexible approach, particularly suitable for sparse content. Despite the effectiveness of LLMs on different tasks, we show that current models struggle with generating structured outputs. In response, we present effective prompting strategies for both of these tasks. We introduce a taxonomy of problems around factuality, global and local structure, common to both modalities and propose a set of critiques to tackle these issues resulting in an absolute improvement in accuracy of $+37$pp (79%) for mind maps and $+15$pp (78%) for tables. To evaluate semantic coverage of generated structured representations we propose Auto-QA, and we verify the adequacy of Auto-QA using SQuAD dataset. We further evaluate the usefulness of structured representations via a text comprehension user study. The results show a significant reduction in comprehension time compared to text when using table (42.9%) and mind map (31.9%), without loss in accuracy.

1 Introduction

The overwhelming amount of information available online poses a significant challenge for users seeking to quickly grasp and process relevant information. Current large language models (LLMs), such as PaLM-2 (PaLM2, 2023), Gemini (Gemini Team, 2023) and ChatGPT (OpenAI, 2022), while capable of providing text-based responses to user queries, often fail to adequately structure and organize this information in a way that facilitates comprehension (Tang et al., 2023). This can lead to information processing bottlenecks that hinder users' ability to efficiently extract meaningful insights from text.

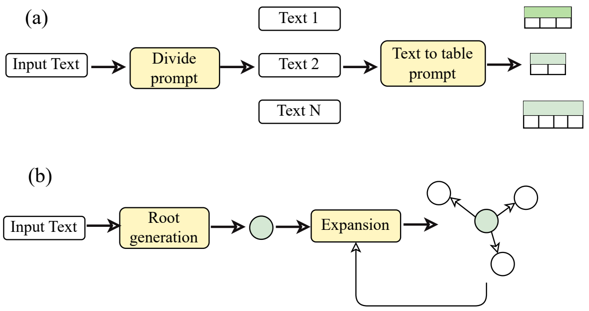


Figure 1: Overview of (a) tables and (b) mind map generation prompts. The prompting steps are colored. Figure (a) illustrates the divide-and-generate prompt. The input passage is initially segmented into sub-passages, followed by the generation of multiple tables. Figure (b) demonstrates the generation process for mind maps. After the main concept has been generated, an iterative expansion phase ensues, during which the mind map is expanded until termination.

To address this issue, we introduce the notion of structured summaries, or StructSum in short. StructSums are derived by hierarchically organizing information and inducing semantic connections from an input text passage. Without loss of generality, we focus on tables (Wu et al., 2022; Li et al., 2023) and mind maps (Buzan, 1996; Huang et al., 2021) as possible StructSum instantiations:

- **Tables** are well-studied in the NLP literature. However the vast majority of the work focused on simpler tasks where tables are inputs – such as QA (Herzig et al., 2020), semantic parsing (Bogin et al., 2019), NLG (Andrejczuk et al., 2022; Puduppully and Lapata, 2021; Laha et al., 2020), etc. – rather than outputs. Indeed, faithfully transforming an arbitrary text passage into a table is a difficult task as the model must deal with different challenges, such as reasoning at multi-

[*] Work done while interning at Google DeepMind.

ple levels, dealing with missing information, and visually consistent formatting. Motivated by the limitations above, we propose to generate multiple tables instead. We argue that this is a simpler task for an LLM, as shown in Figure 2, which compares single-table and multi-table generation side by side. We therefore propose a divide-and-generate prompting approach (see Figure 1) that first divides the input text into multiple text passages, each representing a sub-topic, followed by an LLM prompt to generate a table-caption pair for each smaller passage. This decomposition allows the model to generate smaller, focused and more informative tables, especially for complex text passages with multiple sub-topics.

- **Mind maps** (Hu et al., 2021; Wei et al., 2019) are less studied in the literature, but are helpful for comprehension and learning (Buzan, 1996; Dhindsa et al., 2011). Mind maps are complementary to tables in their structure, allowing for more flexibility and dynamism than tables, as they are inherently schema-less. However, generating mind maps with LLMs presents several challenges: (i) the model first need to select a central concept, that is the fulcrum of all the successive extractions, as mind maps revolve around a central root node; (ii) being a schema-less abstraction, each connecting branch has its own independent sub-topic, making it difficult to automatically add branches all at once; (iii) to ensure readability and well-structuredness each leaf node should terminate the path in a way that concludes the idea or sub-topic; (iv) depending on the information density, some paths may be shorter than others. Therefore, the model should decide whether or not a branch is worth expanding. Following the structure of these observations, we propose an iterative prompting technique for mind map generation. As show in Figure 1, we initialize the mind map by generating the root concept. At each iteration, we decide either to expand the current mind map further or stop the process. During the expansion step, we prompt the model to add branches to the current leaf nodes. We represent the mind map as a JSON object, as it is easy to parse and verify.

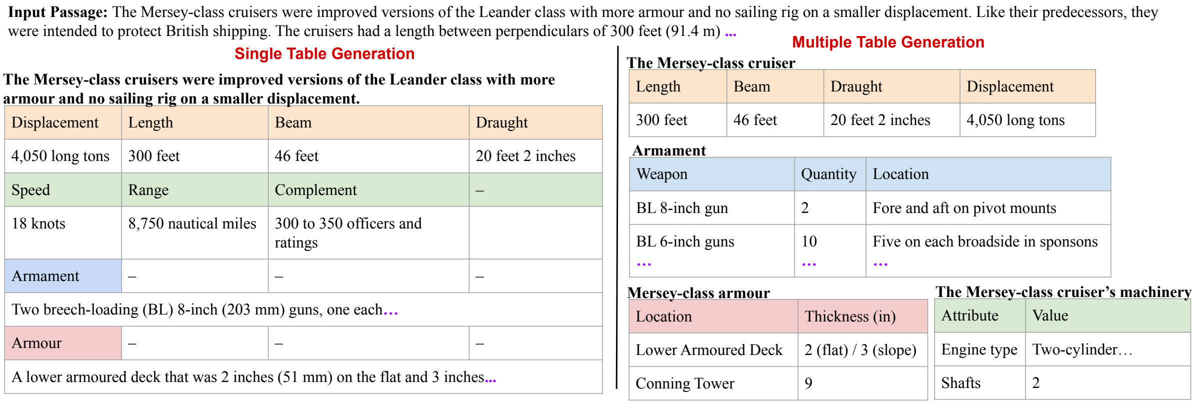
Through extensive experimentation with PaLM-2 (PaLM2, 2023), we show that LLMs are not always effective at generating StructSums that are factual and structurally correct. To overcome these issues we propose a pipeline for structured data generation. Our pipeline consists of structure-specific prompts followed by critics to assess output quality along three different dimensions, that are common both to tables and mind maps: (i) *Factuality*, (ii) *Local Structure* and (iii) *Global Structure*. We found that our proposed critics improved the overall quality of the generated output by +37pp for mind maps and +15pp for tables.

To ensure the usefulness of StructSum for text-comprehension tasks, we propose Auto-QA as a measure of output coverage. We automatically generate QA pairs from input text and use structured outputs to answer these questions. Furthermore, we verify the appropriateness of using Auto-QA by comparing Auto-QA with human generated QA pairs on SQuAD (Rajpurkar et al., 2016) development set.

Finally, starting from the initial hypothesis that StructSums can enhance the effectiveness of information-seeking scenarios, we conducted a user study to evaluate their impact on users' ability to process information, using a text comprehension user study. Results demonstrate how StructSums improve information seeking, specifically on timed text comprehension metrics. We found that by using the structured representation, users can answer questions 42.9% faster for tables and 31.9% for mind maps.

2 Related Work

Structured Output. Generating structured output from text has been explored in the context of information extraction (Li et al., 2023; Pietruszka et al., 2022). Most of the work focus on text-to-table (Wu et al., 2022) generation using the model trained on domain specific dataset. Ni and Li (2023) use LLM for information extraction by generating key-value pairs. Tang et al. (2023) evaluate different models on table generation from text by prompting where table structure is provided as format instructions. Mind map generation has been explored in the form of relation graph structure (Hu et al., 2021; Wei et al., 2019) to summarize new articles (Cheng and Lapata, 2016; Hermann et al., 2015). In contrast, we focus on a generation pipeline applicable for multiple structured outputs types by prompting LLM given a text input. We keep the output structure flexible and domain independent by not instructing the model with specific format.



Input Passage: The Mersey-class cruisers were improved versions of the Leander class with more armour and no sailing rig on a smaller displacement. Like their predecessors, they were intended to protect British shipping. The cruisers had a length between perpendiculars of 300 feet (91.4 m) ...

Single Table Generation

The Mersey-class cruisers were improved versions of the Leander class with more armour and no sailing rig on a smaller displacement.

Displacement	Length	Beam	Draught
4,050 long tons	300 feet	46 feet	20 feet 2 inches
Speed	Range	Complement	–
18 knots	8,750 nautical miles	300 to 350 officers and ratings	
Armament	–	–	–
Two breech-loading (BL) 8-inch (203 mm) guns, one each...			
Armour	–	–	–
A lower armoured deck that was 2 inches (51 mm) on the flat and 3 inches...			

The Mersey-class cruiser

Length	Beam	Draught	Displacement
300 feet	46 feet	20 feet 2 inches	4,050 long tons

Armament

Weapon	Quantity	Location
BL 8-inch gun	2	Fore and aft on pivot mounts
BL 6-inch guns ...	10 ...	Five on each broadside in sponsons ...

Mersey-class armour

Location	Thickness (in)
Lower Armoured Deck	2 (flat) / 3 (slope)
Conning Tower	9

The Mersey-class cruiser's machinery

Attribute	Value
Engine type	Two-cylinder...
Shafts	2

Figure 2: Example table generation for the text at top, comparing single table (left) vs multiple table generation (right). Some parts in the table and text were truncated (...) for readability. The full example is reported in Figure 6.

Prompting. Our prompting strategy is rooted in task decomposition techniques. Least-to-most (Zhou et al., 2023), in contrast with chain-of-thought (Wei et al., 2022), progresses from easiest to hardest questions eventually answering the complete question, while successive prompting (Dua et al., 2022) iteratively generate new questions based on previous answers. Unlike least-to-most, decomposed prompting (Khot et al., 2023) doesn't restrict task decomposition from easiest to the hardest and iteratively generate next steps that can be executed by different systems. Most of the prior work is focused on reasoning for solving QA type problems, in contrast, we are interested in transforming text to structured formats. Our divide-and-generate prompting for multiple table generation (similar to least-to-most) uses an initial prompt to divide the input passage into different topical sub-passages that simplifies the table generation in next step. Different from these tasks our iterative prompting for mind maps requires reasoning over current structured output at each step.

Factuality. Attribution is used as a tool for assessing the reliability of LLMs and identifying potential sources of inaccuracy or fabrication in their generated outputs. Current work apply attribution on unstructured text generation settings, such as, question answering (Bohnet et al., 2022) and text generation tasks (Gao et al., 2023a). Diverging from that, our work require verifying the factuality of generated structured outputs.

Evaluation. Due to the cost of human evaluation, LLMs are used to critique the generated outputs (Wang et al., 2023). Recent instructions tuned models, such as, GPT-4 (OpenAI, 2023) and ChatGPT (OpenAI, 2022) are shown to be strong evaluators. To avoid using external APIs, Kim et al. (2023); Wang et al. (2023) fine-tune a smaller pretrained model to critic model responses. We are interested in evaluating the quality structured outputs using critics and self-correct based on the feedback. As a part of data generation pipeline, our focus is on filtering instances that are incomplete and are not factually grounded.

3 Generating StructSums

We focus on tables and mind maps as a possible StructSum instantiations.

3.1 Tables: Divide & Generate prompting

Given an input text we would like to transform it into multiple tables. Although generating multiple tables from text may seem unnecessary, single-table generation lead to several issues, as shown in Figure 2 (bottom left). The model often produces complex table structures, resulting in missing cell values or the exclusion of relevant information. Additionally, complex tables are difficult to verify for factual accuracy and can require additional mental effort from the user to understand.

To address these limitations, we propose a divide-and-generate approach that dynamically partitions the passage into smaller subtopic segments. While deterministic rule-based chunking methods (e.g., based on word or sentence count) can be employed, they often produce suboptimal results due to potential under-chunking, over-chunking, and the absence of division for certain instances. Therefore, the chunking must be adaptive and depend on the input text and its sub-topic distribution. We use a one-shot prompt for this step, as shown in

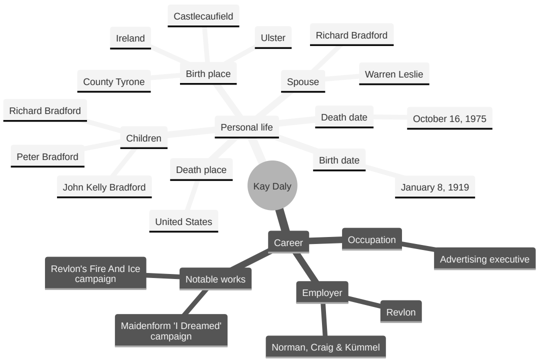


Figure 3: Example mind map output. The full example along with the input text is reported in Figure 7.

Appendix B (Figure 14). After the chunking, we prompt the model to generate a table along with its caption for each sub-passage obtained in the previous step.

Algorithm 1 Mind maps Iterative Prompting

```
Require:
     input text passage: input
     maximum number of steps: max_steps
 1: step ← 0
 2: mindmap ← GENERATE-ROOT(input)
 3: while step < max_steps do
 4:     step ← step + 1
 5:     if CONTINUE-PROMPT(input, mindmap) then
 6:         expansions ← EXPAND(input, mindmap)
 7:         mindmap ← JSON-CRITIC(expansions)
 8:     else
 9:         return mindmap
10:     end if
11: end while
12: return mindmap
```

3.2 Mind maps: Iterative Prompting

Contrary to tables, mind maps (see example in Figure 3) are more flexible and present a different set of challenges. The first challenge is representation. We desire a representation that is (i) close to a familiar format, and (ii) is easily parsable and verifiable using current tools. JSON meets both of these requirements. The second challenge is that mind maps, unlike tables where each row can be produced linearly, necessitate attaching information in different locations depending on which branch is being expanded. This requires the model to think radially.

We propose an iterative prompting for mind maps generation. Algorithm 1 shows the overall procedure. Details of each prompt is in Appendix B. We start by generating the root concept that becomes the central node for the mind map. This separate step allows the model to independently reason about the theme of the passage. After generating the root, at each step we prompt the model to decide if current mind map can be expanded further. If the model decides to expand (line 5), we prompt the model using the current mind map to add more branches. Otherwise, the procedure terminates and we return the current mind map. At each expansion step we sample multiple mind maps. Utilizing the fact that JSON verification is cheaper we select the topmost JSON that is parsed correctly. In the rare case, when none of the samples are parsable we call a critic prompt to correct the top JSON (line 7).

4 Data Generation Pipeline

We now present our STRUCTSUM data generation pipeline. Although each STRUCTSUM is seemingly different, we identify three dimensions that are common to both table and mind map modalities: (i) Factuality, (ii) Local Structure, and (iii) Global Structure. We use a set of critics, implemented via prompts, to ensure sufficient quality across each dimension. Through our initial experiments we find that tweaking each critic according to the structure is more helpful.

4.1 Factuality Critic

We use post-attribution (Gao et al., 2023a) to verify factuality, as we found that jointly generating and attributing (Gao et al., 2023b) results in (i) unnatural text output and (ii) in the model copying verbatim from the input text passage.

Critic cost is one aspect that requires consideration. For example, for tables, verifying each cell could be more robust, however, it increases the number of LLMs calls (listed in Table 1), from $\mathcal{O}(1)$ to $\mathcal{O}(\#\text{number of cells})$.

For simplicity, we choose a single prompt per STRUCTSUM: for tables we ask the LLM to attribute each row, while for mind maps we ask to attribute each path from root to leaf. We convert the input text passage to a list of sentences and ask the model to cite, following the [x,y] format for attribution, the source sentence(s) where the information can be found or [NA] in case this is not possible. The prompts are reported in Figure 15.

4.2 Local Structure Critic

For tables, a common issue arises from the model misplacing values in incorrect columns. For example, placing "66 years" in the *Birth date* column or

Critic	# LLM calls	
	Tables	Mindmaps
Factuality	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Local Structure	$\mathcal{O}$(#cols)	$\mathcal{O}$(#paths)
Global Structure	NA	$\mathcal{O}(1)$

Table 1: Cost for each critic in terms of #LLM calls as proxy. #cols is number of columns in output table. #paths is the number of paths in a mind map from root node to a terminal node.

an address in the *Company Name* column. To detect such errors, we leverage each column header as a category and verify whether all cell values within that column belong to the same category. For mind maps, we observed that a well-defined terminal node can often represent the entire path leading to it. We use this fact and prompt the model to verify whether the terminal node is a specific value, rather than a general concept. The prompts are reported in Figure 16.

4.3 Global Structure Critic

Global critic allows us to verify the overall structure of the output. This means understanding whether all the information contained in a StructSum makes sense globally.

For tables, we simply verify whether the table is well formatted: e.g. we verify equal number columns in the header and subsequent rows, therefore ignoring semantic content of the table and only focusing on form rather than the content. This is realized via simple heuristics implemented in Python (we do not prompt the model for these).

For mind maps, we used a stricter approach, to ensure that information were semantically valid on a global level. Specifically, we convert the mind map into a familiar format like table of contents (ToC), which we hypothesize is more likely to be seen during the pre-training phase of existing LLMs, and ask the model to check if the ToC is at right level of abstraction. The prompts are reported in Figure 17.

5 Semantic Coverage using Auto-QA

In this section we propose an automatic way to assess the quality and the general usefulness of StructSums introducing Auto-QA coverage as proxy metric. [1] This metric measures the semantic coverage or percentage of questions that are answerable when using a StructSum s, instead of the full text passage t. Formally it is defined as:

$$COV(s) = \frac{1}{|\text{GenQA}(t)|} \sum_{i=1}^{|\text{GenQA}(t)|} \mathbb{1}_{E_{a_i}} \left[Q(s, q_i) \right]$$

where GenQA(x) is a function that generates (q, a) pairs given the input text passage t, $Q(s, q_i)$ is a function that generates an answer given in input a StructSum s and the question q_i, whereas the indicator function $\mathbb{1}_{E_{a_i}}(x)$ asses the answer equivalence between a_i and x. Figure 18 in Appendix B, show all the prompts associated with AutoQA module (Deutsch et al., 2021; Fabbri et al., 2022).

Independently of perceived quality, it is worth noting that this simple metric can be thought as an abstractiveness measure or compression quality for a given StructSum s. A value of 1 indicates no information loss at the expense of no compression/abstraction, whereas a value of 0 indicates theoretically maximum compression at the expense of not providing any useful information. A target value is therefore application specific and must be adjusted accordingly [2].

QA pairs generation GenQA(t) is implemented by prompting the LLM to generate a list of question-answer (QA) pairs conditioned on the input text t. To ensure that the quality of QA pairs is sufficient, after generation, we we apply a three-step procedure. First, we removed duplicate questions via string match. Second, we removed answers if none of the words appeared in the input text, thereby ensuring with reasonable certainty that the answer is grounded in the text without being overly stringent. Third, we performed a cyclic consistency check, where we prompted the model to answer the generated question based on input text.

Question answering We use a simple prompt for function $Q(s, q_i)$. For tables, we convert the table representation to a markdown table format, whereas for mind maps we simply serialize the information as a JSON object.

Answer Equivalence As the model might generate verbose answers, verifying whether two an-

[1] We do not present Auto-QA as a substitute for human evaluations of quality. Instead, we propose AutoQA as a coverage metric that allows us to use synthetic question generation.

[2] It is possible to include coverage as a critic. But we opted not to do so, as the threshold for coverage depends on the specific use case. This also allowed us to analyze coverage independently, without being influenced by other factors.

swers are the same is a problem of semantic similarity. Instead of using lexical matching, that is $\mathbb{1}_{E_{a_i}}(x) := a_i = x$, we prompt the model to check if two answers are equivalent.

6 Model

For all the experiments, we use the Unicorn (PaLM-2-unicorn, 2023) variant of PaLM-2, a fine-tuned transformer-based model with UL2 (Tay et al., 2022) like objectives. PaLM-2 improves on PaLM (Chowdhery et al., 2023) through optimized scaling, richer training data and instructing tuning (Wei et al., 2021; Chung et al., 2022).

7 Dataset

To test our pipeline on a diverse set of input passages, we selected Wikipedia text as the source. Specifically, we started with the English split of the Wiki40B (Guo et al., 2020) dataset [3]. The dataset is cleaned up by page filtering to remove disambiguation pages, redirect pages, deleted pages, and non-entity pages. Input to our prompts are passages that are obtained by splitting the Wikipedia text using the _START_PARAGRAPH_ symbol that is already provided as part of the dataset.

7.1 Filtering for Tables Generation

Not all input paragraphs are well suited for table generation. As a proxy for selecting adequate passages, we used regex-based filters to only include passages with more that 20 numeric values and removed passages with less than three sentences. In a real world setting, we would like a systematic way of deciding which modality is adequate for a given text. We leave this exploration as future work.

8 Results

In this section, we present the results of our experiments using PaLM-2.

8.1 Quality impact of prompting style and automated critics

We assessed the quality of generated structured data through manual human ratings. The study was conducted on 100 instances for mind maps. For multi-table generation, we choose 100 individual table-text pair for annotation [4]. Input passages were obtained via data filtering strategy described in Section 7.

Tables		Mindmaps	
Single Table	54	CoT	39
Multi Table	63	Iterative	42

Table 2: Table / mind map accuracy per prompt style. Outputting multiple tables provides higher quality for the table modality. For mindmaps, an iterative approach is to be preferred to a CoT approach. Full prompts are reported in the Appendix B.

Critic	Tables	Mind maps
Baseline†	63	42
↪ Structure	70	71
↪ Factuality	78	79

Table 3: Human annotation accuracy at different pipeline stages. The use of critics is a critical step to improve perceived quality. Local and Global Structure critic provides a significant lift for mind maps. The increase in performance for Factuality, is similar for both Tables and mind map.

Guidelines Annotators were asked to rate each instance as “Good“ or “Bad” by checking the overall quality of the output. For both modalities, annotators were asked to check for factuality as well as the structural quality of the output. To help the annotators measure the structural quality we asked the annotators to check “table structure”, “table header”, “column header-value match” for tables. For mind maps, they were asked to check “incomplete branches”, “not a good main concept”, “too dense / too sparse” and “wrong edge connections”. We also encouraged the annotators to mark the instance as bad if they find any other issues.

Prompt style Table 2 show the results for both the modalities. For table generation task we find that annotators prefer multiple tables generation outputs compared to single table generation. This can be attributed to the fact that multi-table generation enables the model to generate more concise, focused and informative tables. For mind map, we compare chain-of-thought (Wei et al., 2022) with our proposed iterative generation strategy described

[3] We used the version that is available via the Tensorflow datasets https://www.tensorflow.org/datasets/catalog/wiki40b.

[4] We made sure that the input passages are the same for the different ablations within modalities. For multi-table generation, we choose 100 text-table pairs generated using 52 input passages.

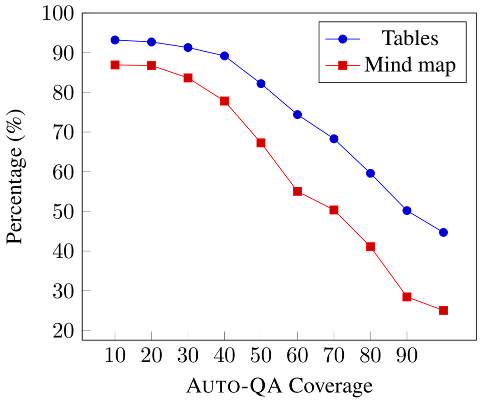


Figure 4: AUTO-QA based coverage. A point $\langle X, Y\rangle$ in each line show that $X\%$ of data has at least $Y\%$ of coverage measured using AUTO-QA.

in Algorithm 1. We find that iterative generation were preferred over simpler prompt outputs.

8.2 Do Critics Align with Human Ratings?

Through our human annotations results in Section 8.1, we find that many generated outputs are not of acceptable quality. To improve the quality of the generated data and to avoid costly human annotations, we propose to use a combination of critics as a measure of data quality. To verify the efficacy of our critics, we first filtered the generated dataset with our critics. Specifically, we performed a logical AND of individual critics and filtered the instances that do not pass the criterion. We then sampled 100 instances from filtered examples and conducted the same evaluation as in Section 8.1.

Results in Table 3 show that using the proposed critics the overall quality is improved by a significant margin. We observe that data filtered using *Structure* (Global and Local) and *Factuality* critics improve the percentage of acceptable instances generated using the pipeline. We find that the quality of mind maps improve by absolute $+37pp$. Similarly, for tables quality improves by absolute $+15pp$. These results indicate that the critics were able to retain good examples and that the selection criterion is in agreement with human judgement.

8.3 Measuring Coverage via Auto-QA

Results in Figure 4 show AUTO-QA coverage for mind maps and tables. The curve shows for a particular coverage threshold what percentage of data meets that threshold. Overall, we observe that tables have better coverage compared to mind maps, meaning that they have an higher abstractiveness or information retention capacity. Interestingly, even though both modalities are perceptually different, we notice that both of them follow similar trends.

	QA Type	
	Auto	Human
Mind map	55.6	61.4
Multi-Table (Divide-and-generate)	66.8	69.3
Single Table	57.1	58.8
Query Focused (Single Table)	81	85.5

Table 4: QA accuracy on different modalities as context, generated using SQuAD validation set. AUTO-QA is automatic question-answer pair generation. Human QA are original SQuAD questions curated by humans.

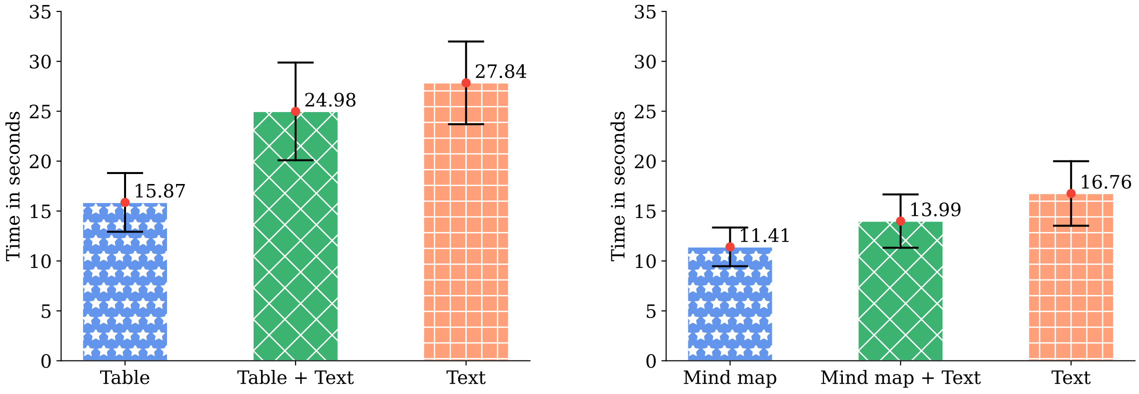
8.4 Is Auto-QA a reasonable metric?

We investigate the feasibility of using AUTO-QA as a surrogate for manually written QA pairs. We aim to determine whether AUTO-QA can generate QA pairs of comparable quality to those written by humans, and leading to a similar evaluation of semantic coverage. To verify the same, we use randomly selected 1000 <*passage, question, answer*> triples from the SQuAD (Rajpurkar et al., 2016) validation set (common for all the experiments). Using the text *passage* as input we generate different STRUCTSUMS. Next, we generate a QA pair corresponding to each text *passage*. This QA pairs acts as a substitute for human written QA pair for AUTO-QA study. The goal is to check whether, keeping the passage and output STRUCTSUM the same, there is a correlation in performance between human generated QA pairs and automatically generated QA pairs.

Table 4 shows the overall results. Second (Mind maps) and third (Multi-Table) row show the comparison between Human QA and Auto QA for our proposed divide-and-generate prompt for tables and iterative prompt for Mindmap generation. We can see that AUTO-QA has comparable results and is a reasonable substitute for human generated questions as a measure of semantic coverage. We further study the limitations of AUTO-QA, and the difference in Human vs Auto QA scores in Section 10.

8.5 Multiple Tables vs Single Table

To check whether generating multiple tables is better at covering more information, we perform a comparison between the ability to answer questions by generating single or multiple tables. On



(a) Table results

(b) Mind map results

Figure 5: Results for timed text comprehension based user study. Plots show 95% confidence interval over time taken in seconds to answer question with different structure combinations as context. For both tables (left) and mind map (right), compared to text only, we observe significant reduction (42.9% and 31.9% resp.) in average time taken by annotators to answer the question.

comparing Multi-Table and Single Table row in Table 4, we observe that for both Auto-QA and Human QA generating multiple table provides more coverage. So in addition to the benefits such as comparatively better verifiability and robust generation, multiple table generation are also better at covering more semantic information.

8.6 Query Focused Generation

In many cases user intent is known in advance, for example, a user query to search or LLM-based Assistant interface (e.g., ChatGPT, Gemini, etc.). We explore the possibility of generating structured data in the presence of a query. We perform a *preliminary* analysis by adding the query in single table generation prompt. As we can see in last row in Table 4, query focused generation improve the performance by more than 20 for Auto-QA and 25 points for Human-QA. Since this requires further investigation in terms of prompting and output quality analysis, we leave a comprehensive exploration of query-focused structured data generation as future work.

8.7 Are StructSums useful?

We evaluate whether StructSum are useful abstractions for the users. For this we design a *timed* text comprehension based user study. We assume that user is looking to answer a specific query, i.e. has a specific intent. We measure time taken to satisfy the user intent as a proxy of usefulness. Our evaluation team consists of 12 volunteers who are affiliated with our institution. Five of these volunteers are female, and seven are male. All volunteers are proficient in English, although not native speakers.

We create an intent in the form of a question along with different context combinations. For example, for a question q, we create $\langle q, s \rangle$, $\langle q, t \rangle$, $\langle q, s + t \rangle$ as possible combinations, where s is a StructSum and t is the input text passage. Each of these combinations are presented to different annotators while ensuring that no annotator see the same question twice. We then measure how long it takes to answer the question in each scenario.

StructSums for the study were generated using our data generation pipeline and critic-based filtering, as discussed in Section 4. In total 600 instances were annotated, equally divided into different context combinations for mind maps and table generation. Annotators consistently answered correctly across all context combinations (Appendix A), suggesting that the level of context did not significantly impact their accuracy. Figure 5 shows the overall results. The plots show 95% confidence interval of time taken by the annotators when using different modality:

- **Tables.** Figure 5a shows that on average annotators with access to tables were able to answer almost 42.9% time faster on average compared to annotators with only text. Furthermore, we observe that presenting both table and text is also useful to the annotators.
- **Mind maps.** Figure 5b shows the results for

	QA Type	
	Auto	Human
Mind map	60.2 [57.0, 63.3]	61.3 [58.1, 64.4]
Multi-Table (Divide-and-generate)	72.3 [69.4, 75.2]	68.9 [66.0, 71.9]
Single Table	61.8 [58.7, 64.9]	58.1 [54.9, 61.3]
Query Focused (Single Table)	87.7 [85.5, 89.8]	85.5 [83.2, 87.8]

Table 5: Human and Auto-QA accuracy with 95% confidence interval on different modalities as context. Unlike the results presented in Table 4, we have excluded SQuAD passages for which none of the questions generated by Auto-QA passed the filter.

Tables	
Avg #words per chunk	114.8
Avg #sentences per chunk	3.9
Avg #words per input	240.6
Avg #sentences per input	8.1
Avg #rows	7.1
Avg #cols	3.3
Avg #tables	1.9
Max #tables	11
Mind map	
Avg #words	194.6
Avg #sentences	7.9
Avg #nodes	11.8
Avg depth	2.2

Table 6: Table / mind map text input and output statistics. On average two ($\sim$ 1.9) tables (top) are generated per input text instance. Mind maps (bottom) contains 11.8 nodes on average.

the study with mind map. A similar trend can be observed, with a reduction of approximately 31.9% in average time between annotators with mind maps compared to annotators that only used text to answer the question.

We note that $\langle q, s + t\rangle$ performs worse than $\langle q, s\rangle$. We believe this is due to the fact that the annotators cross-checked the answer from both the modalities, leading to increase in time to answer the question.

9 Data Generation statistics

Table 6 shows different statistics of data generated using our prompts. For tables generation we observe that our methods generate almost two ($\sim$ 1.9) tables per instance and the tables have 7.1 rows and 3.3 columns on average. Mind maps have an average of 11.8 nodes with a depth of 2.2. We show example mind map and table generation in Figure 7 and Figure 6 respectively.

10 Limitations of Auto-QA

We propose Auto-QA as a coverage metric, in order to measure how much information from the original passage is retained in the StructSum.

Note that the metric is not intended as a substitute for evaluations of overall quality. The primary benefit of Auto-QA is the synthetic question generation component, which can be run at scale without costly human annotations. In Table 4, we measured how closely the synthetic Auto-QA aligns with human generated question-answer pairs, revealing certain limitations of the metric. Notably the quality of generated questions is not always reasonable, which is mitigated using specialized filters, as discussed in Section 5. Although we generate several question-answer pairs, it is possible that none pass the filters, which we see for fewer than 10% of input passages. Such instances adversely affect the Auto-QA coverage score, contributing to the score discrepancies in Table 4. When the analysis is restricted to those subsets where the Auto-QA filtering is successful, the observed differences are diminished and fall within the bounds of experimental noise, as reported in Table 5.

11 Conclusion

In this work we study the potential of structured representations like tables and mind maps to enhance information comprehension. Utilizing our divide-and-generate prompting and iterative expansion, we achieved significant improvements in output quality (+$37pp$ for mind maps, +$15pp$ for tables) using structure-specific prompts and critics. We proposed Auto-QA based coverage metric that automatically generates QA pairs from the input text and uses StructSum outputs to answer them.

12 Acknowledgment

We would like to thank Massimo Nicosia and Yasemin Altun for their feedback and comments on the paper. We are grateful to all the volunteers who participated in our annotation experiments.

13 Limitations

We outline the limitations of our work to ensure transparency and inspire future research. First, the structured output representations we experimented with are limited to tables and mind maps. However, to comprehensively evaluate the effectiveness of our critics and pipeline, it is desirable to also evaluate other input and output modalities, e.g. image and video, considering the recent advances in VLMs. Secondly, our work and experimental findings are limited to only English sources. We plan to also explore multilingual structured summaries in future work. Third, we would to warn against the risk of blindly trusting models to generate structured summaries from an input accurately. Although we take extra care to increase the factuality of the outputs via the use of critics, and experimentally validate QA coverage, we believe that special care should be taken to verify outputs in accuracy-sensitive applications. Finally, our STRUCTSUM generation is performed using a LLM with fixed prompts, however, prior work have shown a reasonable portability of prompts across similar models (Zhou et al., 2023; Khot et al., 2023). Despite these limitations, our work serves as an initial step in constructing reliable structured summarization evaluations, models and applications. We hope future research can greatly benefit from this starting point.

References

Ewa Andrejczuk, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, and Yasemin Altun. 2022. Table-to-text generation and pre-training with TabT5. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6758–6766, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-SQL parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3659–3664, Hong Kong, China. Association for Computational Linguistics.

Bernd Bohnet, Vinh Q. Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Massimiliano Ciaramita, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, Kai Hui, Tom Kwiatkowski, Ji Ma, Jianmo Ni, Lierni Sestorain Saralegui, Tal Schuster, William W. Cohen, Michael Collins, Dipanjan Das, Donald Metzler, Slav Petrov, and Kellie Webster. 2022. Attributed question answering: Evaluation and modeling for attributed large language models. *arXiv preprint*.

Tony Buzan. 1996. *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. Plume.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Daniel Deutsch, Tania Bedrax-Weiss, and Dan Roth. 2021. Towards Question-Answering as an Automatic Metric for Evaluating the Content Quality of a Summary. *Transactions of the Association for Computational Linguistics*, 9:774–789.

Harkirat S. Dhindsa, Makarimi-Kasim, and O. Roger Anderson. 2011. Constructivist-visual mind map teaching approach and the quality of students' cognitive structures. *Journal of Science Education and Technology*, 20(2):186–200.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alexander Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. QAFactEval: Improved QA-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601, Seattle, United States. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023a. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023b. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, Singapore. Association for Computational Linguistics.

Gemini Team. 2023. Gemini: A family of highly capable multimodal models. Technical report, Google.

Mandy Guo, Zihang Dai, Denny Vrandecic, and Rami Al-Rfou. 2020. Wiki-40b: Multilingual language model dataset. In *LREC 2020*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Mengting Hu, Honglei Guo, Shiwan Zhao, Hang Gao, and Zhong Su. 2021. Efficient mind-map generation via sequence-to-graph and reinforced graph refinement. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8130–8141, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. *Preprint*, arXiv:2310.08491.

Anirban Laha, Parag Jain, Abhijit Mishra, and Karthik Sankaranarayanan. 2020. Scalable Micro-planned Generation of Discourse from Structured Data. *Computational Linguistics*, 45(4):737–763.

Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. 2023. A sequence-to-sequence&set model for text-to-table generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5358–5370, Toronto, Canada. Association for Computational Linguistics.

Xuanfan Ni and Piji Li. 2023. Unified text structuralization with instruction-tuned language models. *arXiv preprint arXiv:2303.14956*.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

PaLM-2-unicorn. 2023. PaLM-2 google ai blog. `https://blog.google/technology/ai/google-palm-2-ai-large-language-model/`. Accessed: 2023-05-10.

PaLM2. 2023. PaLM2 technical report. `https://ai.google/static/documents/palm2techreport.pdf`. Accessed: 2023-05-10.

Michał Pietruszka, Michał Turski, Łukasz Borchmann, Tomasz Dwojak, Gabriela Pałka, Karolina Szyndler, Dawid Jurkiewicz, and Łukasz Garncarek. 2022. Stable: Table generation framework for encoder-decoder models. *arXiv preprint arXiv:2206.04045*.

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text Generation with Macro Planning. *Transactions of the Association for Computational Linguistics*, 9:510–527.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2023. Struc-bench: Are large language models really good at generating complex structured data? *Preprint*, arXiv:2309.08963.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, et al. 2022. Ul2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yang Wei, Honglei Guo, Jinmao Wei, and Zhong Su. 2019. Revealing semantic structures of texts: Multi-grained framework for automatic mind-map generation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5247–5254. International Joint Conferences on Artificial Intelligence Organization.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2022. Text-to-table: A new way of information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2518–2533, Dublin, Ireland. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A User Study

Usually, mind maps are represented as a graph as shown in Figure 7. However, for the text comprehension user study described in Section 8.7, to avoid bias due to color or orientation, we simplify the representation as a tree (Figure 10). To establish the known query intent, annotators' are first shown with input question, e.g., Figure 8. Next, on clicking `Show content` button, annotators are shown context in the form of either text (Figure 9), structure (Figure 10), or structure + text (Figure 11). The question-answer pairs were generated automatically conditioned on input text (Section 5). Annotators were also allowed to mark an instance un-answerable. The user study for tables is performed in a similar manner. We annotated 100 question-answer pairs for both mind maps and tables. Each input instance is annotated with three different context combinations, leading to 600 total annotations. We filtered instances that were marked un-answerable by the annotators (32% and 22% for tables and mind map study resp.). To avoid penalizing for spelling errors or other typing mistakes, the answers were evaluated via human evaluation.

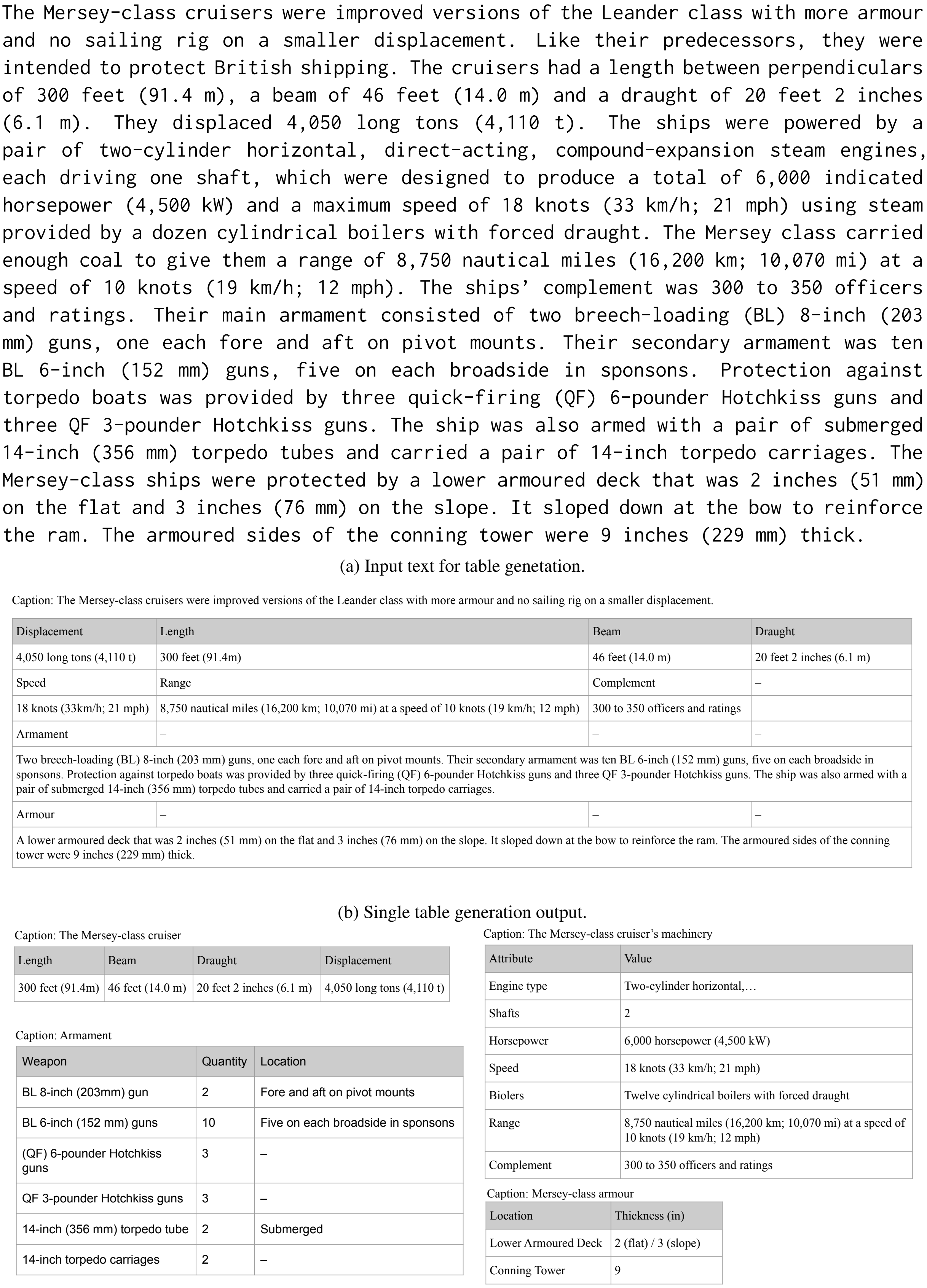
We adopt timed-comprehension for answering free-form questions as a proxy measure for usefulness of generated structured representation. This makes it different from categorical data annotations. We calculate rater agreement for the questionnaire responses. We find that 89.9% of questions had full rater agreement regarding the correct response, with the only differences in the time taken to respond. Each question was shown to three raters. Table 7 shows the overall accuracy as percentage of questions answered correctly in different context. Irrespective of context combinations, annotators were able to answer the questions correctly with a high accuracy.

Tables		**Mindmaps**	
Table	95.6	Mind map	97.7
Text	94.1	Text	94.3
Table+Text	94.1	Mind map+Text	97.7

Table 7: Answer accuracy (as percentage) for different context combinations. Structure context performs on par/better compared to text.

B Prompts

We include the different prompts used in this study. In our implementation we use Jinja (`https://jinja.palletsprojects.com/`) to specify the prompt template. The prompts can be found in Figures 12 to 18.



The Mersey-class cruisers were improved versions of the Leander class with more armour and no sailing rig on a smaller displacement. Like their predecessors, they were intended to protect British shipping. The cruisers had a length between perpendiculars of 300 feet (91.4 m), a beam of 46 feet (14.0 m) and a draught of 20 feet 2 inches (6.1 m). They displaced 4,050 long tons (4,110 t). The ships were powered by a pair of two-cylinder horizontal, direct-acting, compound-expansion steam engines, each driving one shaft, which were designed to produce a total of 6,000 indicated horsepower (4,500 kW) and a maximum speed of 18 knots (33 km/h; 21 mph) using steam provided by a dozen cylindrical boilers with forced draught. The Mersey class carried enough coal to give them a range of 8,750 nautical miles (16,200 km; 10,070 mi) at a speed of 10 knots (19 km/h; 12 mph). The ships' complement was 300 to 350 officers and ratings. Their main armament consisted of two breech-loading (BL) 8-inch (203 mm) guns, one each fore and aft on pivot mounts. Their secondary armament was ten BL 6-inch (152 mm) guns, five on each broadside in sponsons. Protection against torpedo boats was provided by three quick-firing (QF) 6-pounder Hotchkiss guns and three QF 3-pounder Hotchkiss guns. The ship was also armed with a pair of submerged 14-inch (356 mm) torpedo tubes and carried a pair of 14-inch torpedo carriages. The Mersey-class ships were protected by a lower armoured deck that was 2 inches (51 mm) on the flat and 3 inches (76 mm) on the slope. It sloped down at the bow to reinforce the ram. The armoured sides of the conning tower were 9 inches (229 mm) thick.

(a) Input text for table genetation.

Caption: The Mersey-class cruisers were improved versions of the Leander class with more armour and no sailing rig on a smaller displacement.

Displacement	Length	Beam	Draught
4,050 long tons (4,110 t)	300 feet (91.4m)	46 feet (14.0 m)	20 feet 2 inches (6.1 m)
Speed	Range	Complement	–
18 knots (33km/h; 21 mph)	8,750 nautical miles (16,200 km; 10,070 mi) at a speed of 10 knots (19 km/h; 12 mph)	300 to 350 officers and ratings	
Armament	–	–	–
Two breech-loading (BL) 8-inch (203 mm) guns, one each fore and aft on pivot mounts. Their secondary armament was ten BL 6-inch (152 mm) guns, five on each broadside in sponsons. Protection against torpedo boats was provided by three quick-firing (QF) 6-pounder Hotchkiss guns and three QF 3-pounder Hotchkiss guns. The ship was also armed with a pair of submerged 14-inch (356 mm) torpedo tubes and carried a pair of 14-inch torpedo carriages.			
Armour	–	–	–
A lower armoured deck that was 2 inches (51 mm) on the flat and 3 inches (76 mm) on the slope. It sloped down at the bow to reinforce the ram. The armoured sides of the conning tower were 9 inches (229 mm) thick.			

(b) Single table generation output.

Caption: The Mersey-class cruiser

Length	Beam	Draught	Displacement
300 feet (91.4m)	46 feet (14.0 m)	20 feet 2 inches (6.1 m)	4,050 long tons (4,110 t)

Caption: Armament

Weapon	Quantity	Location
BL 8-inch (203mm) gun	2	Fore and aft on pivot mounts
BL 6-inch (152 mm) guns	10	Five on each broadside in sponsons
(QF) 6-pounder Hotchkiss guns	3	–
QF 3-pounder Hotchkiss guns	3	–
14-inch (356 mm) torpedo tube	2	Submerged
14-inch torpedo carriages	2	–

Caption: The Mersey-class cruiser's machinery

Attribute	Value
Engine type	Two-cylinder horizontal,…
Shafts	2
Horsepower	6,000 horsepower (4,500 kW)
Speed	18 knots (33 km/h; 21 mph)
Biolers	Twelve cylindrical boilers with forced draught
Range	8,750 nautical miles (16,200 km; 10,070 mi) at a speed of 10 knots (19 km/h; 12 mph)
Complement	300 to 350 officers and ratings

Caption: Mersey-class armour

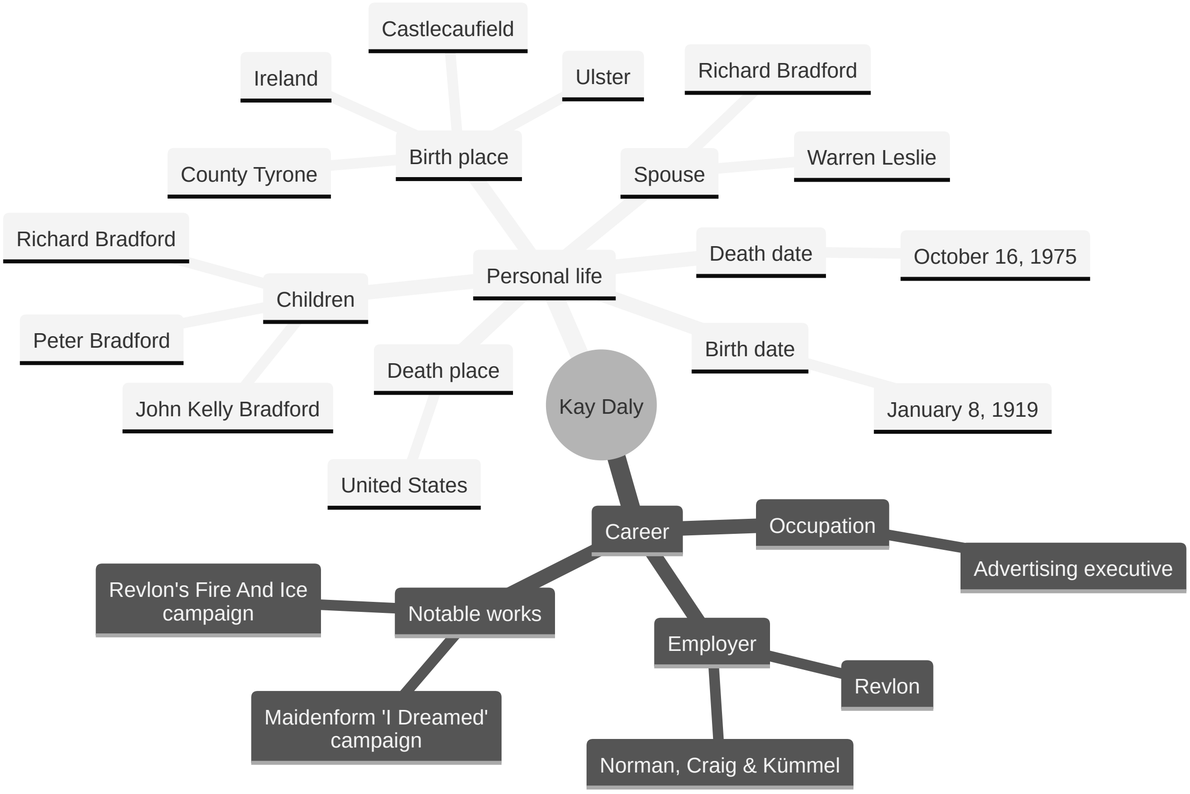
Location	Thickness (in)
Lower Armoured Deck	2 (flat) / 3 (slope)
Conning Tower	9

(c) Multiple table generation output.

Figure 6: Example outputs for single and multiple table generation approach. Text in (a) shows the input. (b) and (c) show the outputs for single and multiple table generation respectively.

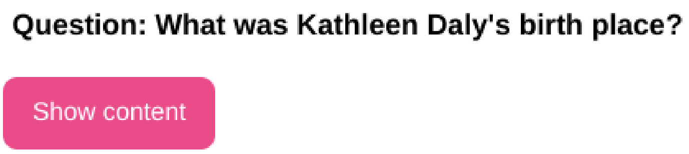
Kathleen "Kay" Daly (January 8, 1919 – October 16, 1975) was an Irish-born American advertising executive and one of the four ”celebrated Daly sisters”. At Norman, Craig & Kümmel she was the creative force behind the famous Maidenform "I Dreamed ...” campaign and Revlon’s legendary 1952 Fire And Ice campaign, working with photographer Richard Avedon. She also was responsible for the line ”Every woman alive loves Chanel Number Five”. She went on to join Revlon in 1961 as vice president and creative director. Kathleen Daly was born in Castlecaufield, County Tyrone, Ulster, Ireland, in 1919. Northern Ireland was created two years later with Tyrone one of its six counties. The family emigrated early in the 1920s. She grew up as one of four sisters, Maggie, Kay, Maureen, and American-born Sheila. They became known for their writing and work in journalism, fashion, and advertising, and were called "the celebrated Daly sisters” by Time magazine in 1966. Life magazine ran a feature story on them in 1949 and a follow-up in 1959. All four were at least once employed by the Chicago Tribune. When she moved to San Francisco after World War II, Kay Daly famously rented space on a billboard to advertise for an apartment. It not only netted her an apartment, but netted her nationwide fame and countless marriage proposals. She had a brief marriage to BMW executive and film producer Richard Bradford (part of the famous Bradford family of Plymouth Colony), who fathered her sons John (Kelly), Richard, and Peter. She then was married to journalist and executive Warren Leslie, who adopted and raised her sons, until her death on October 16, 1975, of pancreatic cancer. She was survived by husband Warren, sons Kelly, Peter, and Richard Bradford, and stepsons Warren and Michael Leslie.

(a) Input text for mind map generation.



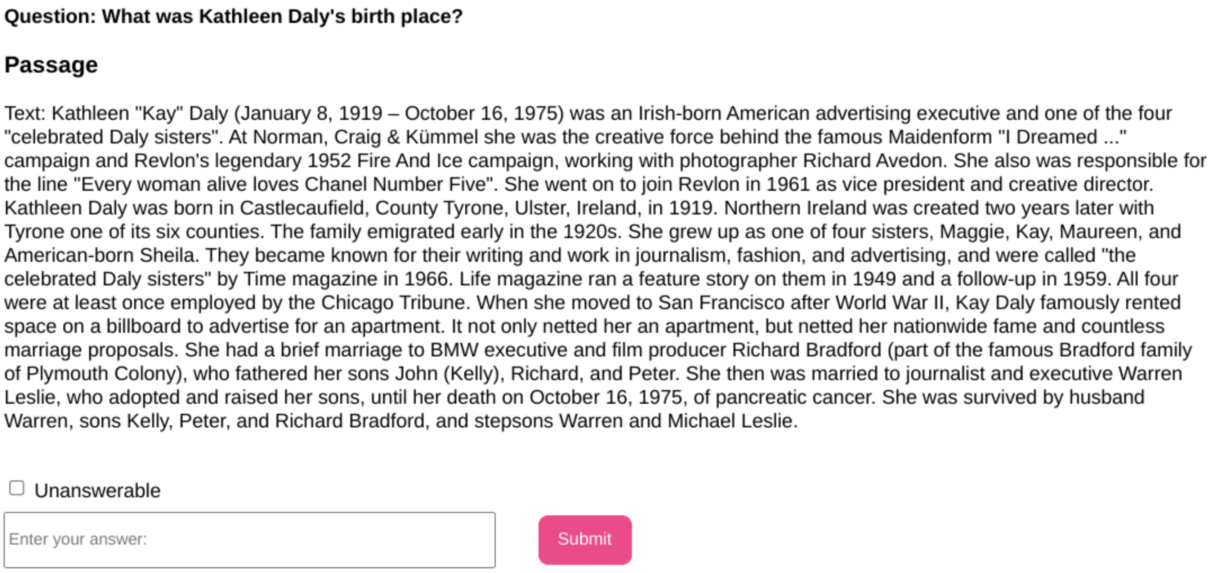
(b) Mind map output.

Figure 7: Example mind map (below) generation for the input text (above). We use mermaid.js (https://mermaid.js.org/) to visualize the output.



Question: What was Kathleen Daly's birth place?

Figure 8: Example UI frame that is shown at the beginning of each annotation instance.

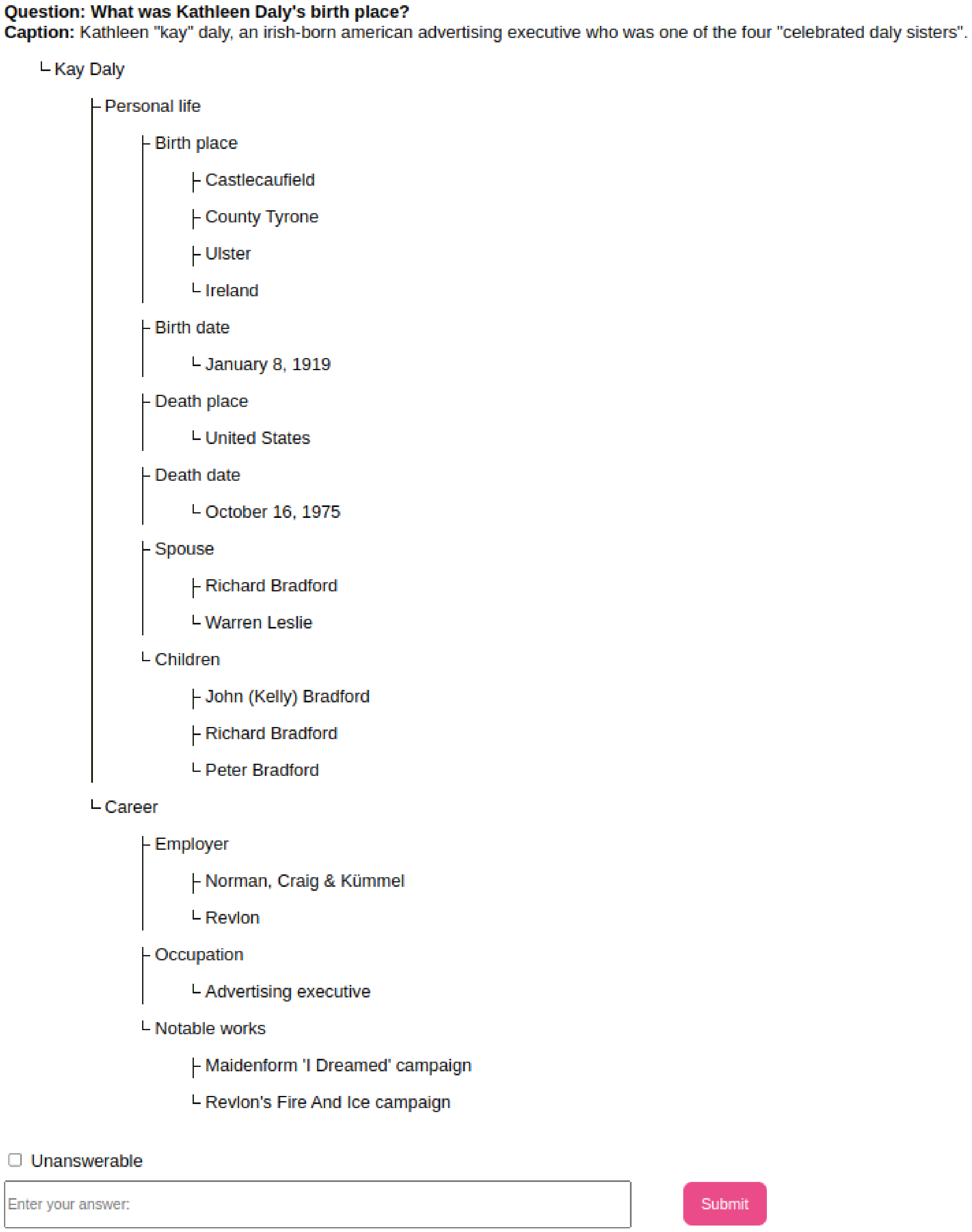


Question: What was Kathleen Daly's birth place?

Passage

Text: Kathleen "Kay" Daly (January 8, 1919 – October 16, 1975) was an Irish-born American advertising executive and one of the four "celebrated Daly sisters". At Norman, Craig & Kümmel she was the creative force behind the famous Maidenform "I Dreamed ..." campaign and Revlon's legendary 1952 Fire And Ice campaign, working with photographer Richard Avedon. She also was responsible for the line "Every woman alive loves Chanel Number Five". She went on to join Revlon in 1961 as vice president and creative director. Kathleen Daly was born in Castlecaufield, County Tyrone, Ulster, Ireland, in 1919. Northern Ireland was created two years later with Tyrone one of its six counties. The family emigrated early in the 1920s. She grew up as one of four sisters, Maggie, Kay, Maureen, and American-born Sheila. They became known for their writing and work in journalism, fashion, and advertising, and were called "the celebrated Daly sisters" by Time magazine in 1966. Life magazine ran a feature story on them in 1949 and a follow-up in 1959. All four were at least once employed by the Chicago Tribune. When she moved to San Francisco after World War II, Kay Daly famously rented space on a billboard to advertise for an apartment. It not only netted her an apartment, but netted her nationwide fame and countless marriage proposals. She had a brief marriage to BMW executive and film producer Richard Bradford (part of the famous Bradford family of Plymouth Colony), who fathered her sons John (Kelly), Richard, and Peter. She then was married to journalist and executive Warren Leslie, who adopted and raised her sons, until her death on October 16, 1975, of pancreatic cancer. She was survived by husband Warren, sons Kelly, Peter, and Richard Bradford, and stepsons Warren and Michael Leslie.

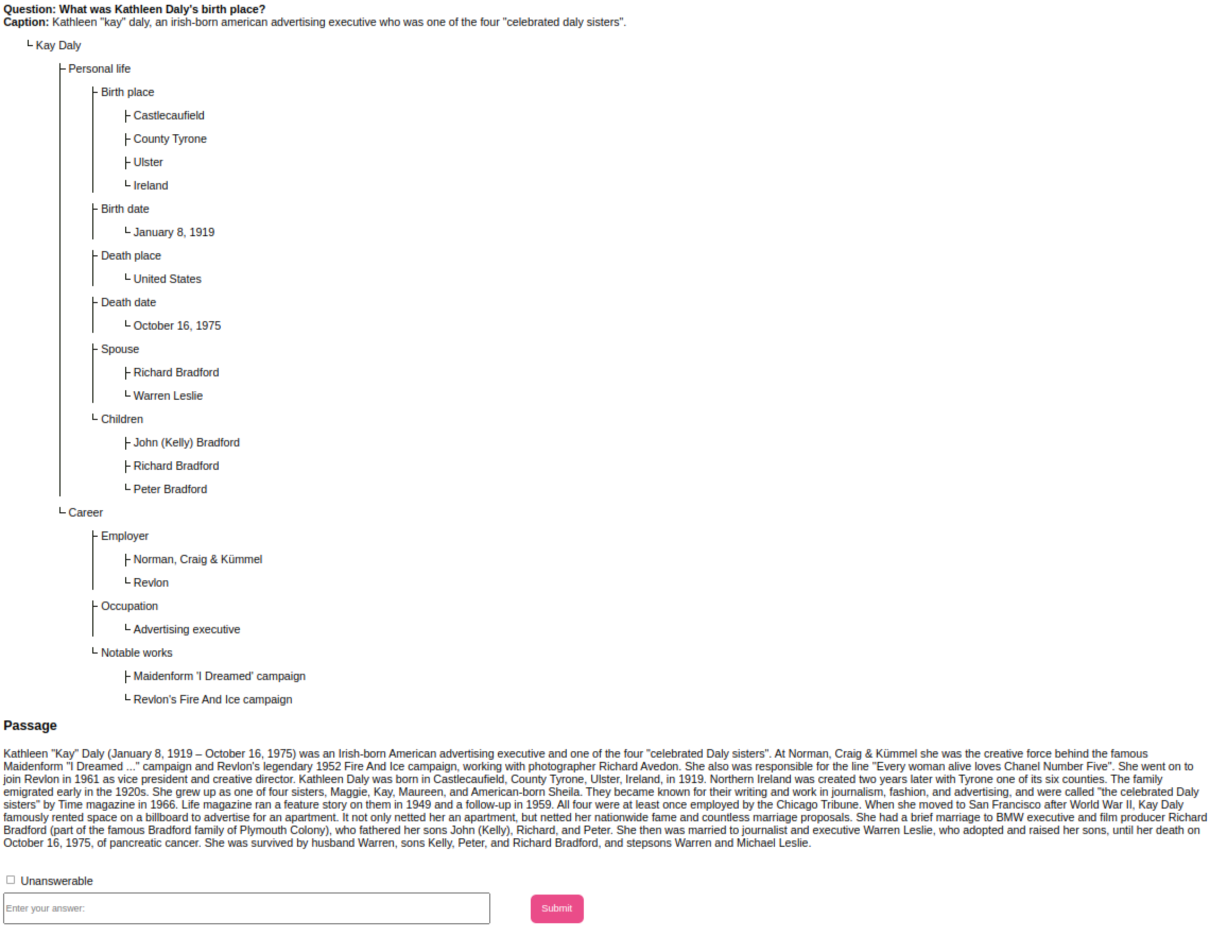
Figure 9: A followup frame shown after Figure 8 with text as context.



Question: What was Kathleen Daly's birth place?
Caption: Kathleen "kay" daly, an irish-born american advertising executive who was one of the four "celebrated daly sisters".

```
└ Kay Daly
    ├ Personal life
    │   ├ Birth place
    │   │   ├ Castlecaufield
    │   │   ├ County Tyrone
    │   │   ├ Ulster
    │   │   └ Ireland
    │   ├ Birth date
    │   │   └ January 8, 1919
    │   ├ Death place
    │   │   └ United States
    │   ├ Death date
    │   │   └ October 16, 1975
    │   ├ Spouse
    │   │   ├ Richard Bradford
    │   │   └ Warren Leslie
    │   └ Children
    │       ├ John (Kelly) Bradford
    │       ├ Richard Bradford
    │       └ Peter Bradford
    └ Career
        ├ Employer
        │   ├ Norman, Craig & Kümmel
        │   └ Revlon
        ├ Occupation
        │   └ Advertising executive
        └ Notable works
            ├ Maidenform 'I Dreamed' campaign
            └ Revlon's Fire And Ice campaign
```

Figure 10: A followup frame shown after Figure 8 with structure (mind map) output as context.



Question: What was Kathleen Daly's birth place?
Caption: Kathleen "kay" daly, an irish-born american advertising executive who was one of the four "celebrated daly sisters".

```
  └ Kay Daly
        ├ Personal life
        │     ├ Birth place
        │     │     ├ Castlecaufield
        │     │     ├ County Tyrone
        │     │     ├ Ulster
        │     │     └ Ireland
        │     ├ Birth date
        │     │     └ January 8, 1919
        │     ├ Death place
        │     │     └ United States
        │     ├ Death date
        │     │     └ October 16, 1975
        │     ├ Spouse
        │     │     ├ Richard Bradford
        │     │     └ Warren Leslie
        │     └ Children
        │           ├ John (Kelly) Bradford
        │           ├ Richard Bradford
        │           └ Peter Bradford
        └ Career
              ├ Employer
              │     ├ Norman, Craig & Kümmel
              │     └ Revlon
              ├ Occupation
              │     └ Advertising executive
              └ Notable works
                    ├ Maidenform 'I Dreamed' campaign
                    └ Revlon's Fire And Ice campaign
```

Passage

Kathleen "Kay" Daly (January 8, 1919 – October 16, 1975) was an Irish-born American advertising executive and one of the four "celebrated Daly sisters". At Norman, Craig & Kümmel she was the creative force behind the famous Maidenform "I Dreamed ..." campaign and Revlon's legendary 1952 Fire And Ice campaign, working with photographer Richard Avedon. She also was responsible for the line "Every woman alive loves Chanel Number Five". She went on to join Revlon in 1961 as vice president and creative director. Kathleen Daly was born in Castlecaufield, County Tyrone, Ulster, Ireland, in 1919. Northern Ireland was created two years later with Tyrone one of its six counties. The family emigrated early in the 1920s. She grew up as one of four sisters, Maggie, Kay, Maureen, and American-born Sheila. They became known for their writing and work in journalism, fashion, and advertising, and were called "the celebrated Daly sisters" by Time magazine in 1966. Life magazine ran a feature story on them in 1949 and a follow-up in 1959. All four were at least once employed by the Chicago Tribune. When she moved to San Francisco after World War II, Kay Daly famously rented space on a billboard to advertise for an apartment. It not only netted her an apartment, but netted her nationwide fame and countless marriage proposals. She had a brief marriage to BMW executive and film producer Richard Bradford (part of the famous Bradford family of Plymouth Colony), who fathered her sons John (Kelly), Richard, and Peter. She then was married to journalist and executive Warren Leslie, who adopted and raised her sons, until her death on October 16, 1975, of pancreatic cancer. She was survived by husband Warren, sons Kelly, Peter, and Richard Bradford, and stepsons Warren and Michael Leslie.

☐ Unanswerable

Figure 11: A followup frame shown after Figure 8 with structure (mind map) output and input text as context.

```
{% set step1 = '{"node": "Global Climate change"}' -%}
{% set step2 = '{"node": "Global Climate change","branches
      ↪ ":[{"node": "effects"},{"node": "causes"},{"node":
      ↪ "solutions"}]}' -%}
{% set step3 = '{"node": "Global Climate change","branches
      ↪ ":[{"node": "effects","branches": [{"node": "
      ↪ melting ice"},{"node": "heat waves"}]},{"node": "
      ↪ causes","branches": [{"node": "Enhanced greenhouse
      ↪ effect"},{"node": "Pollution"}]},{"node": "
      ↪ solutions","branches": [{"node": "Individual
      ↪ efforts"},{"node": "International resolutions
      ↪ "}]}]}' -%}
{% set step4 = '{"node": "Global Climate change","branches
      ↪ ":[{"node": "effects","branches": [{"node": "
      ↪ melting ice"},{"node": "heat waves","branches": [{"
      ↪ node" : "droughts"}]}]},{"node": "causes","branches
      ↪ ": [{"node": "Enhanced greenhouse effect"},{"node":
      ↪  "Pollution","branches": [{"node": "Carbon emission
      ↪ "},{"node": "Burning coal"}]}]},{"node": "solutions
      ↪ ","branches": [{"node": "Individual efforts"},{"
      ↪ node": "International resolutions"}]}]}' -%}
A mind map is a diagram used to visually organize
      ↪ information into a hierarchy, showing relationships
      ↪  among pieces of the whole. It is often created
      ↪ around a single concept.  Major ideas are connected
      ↪  directly to the central concept, and other ideas
      ↪ branch out from those major ideas. Mind maps can be
      ↪  generated based on the content present in text in
      ↪ multiple steps.
Consider the following example.
Given the following text:
Global climate change has many effects, including melting
      ↪ ice, heat waves, and droughts. It is caused by the
      ↪ enhanced greenhouse effect, which is caused by
      ↪ pollution, such as carbon emissions and burning
      ↪ coal. Solutions to global climate change include
      ↪ individual efforts and international resolutions.

Choose primary concept that is the root
Output:
MindMap
{{ format_json(step1)|safe }}
END_THOUGHT
Can we add branches ?
Output: Yes
END_THOUGHT
Add branches:
MindMap
{{ format_json(step2)|safe }}
END_THOUGHT
Can we add branches ?
Output: Yes
END_THOUGHT
Add branches:
MindMap
{{ format_json(step3)|safe }}
END_THOUGHT
Can we add branches ?
Output: Yes
END_THOUGHT
Add branches:
MindMap
{{ format_json(step4)|safe }}
END_THOUGHT
Can we add branches ?
Output: No
END_THOUGHT

Now for the text below:
{{input_text}}
Choose primary concept that is the root
Output:
MindMap{%- if root %}
{{root}}
END_THOUGHT
{% if current_mindmap -%}
Can we add branches?
Output: Yes
Add branches:
MindMap
{{current_mindmap}}
END_THOUGHT
{%- endif %}
Can we add branches ?
{%- if y_n_current %}
Output: {{y_n_current}}
END_THOUGHT
Add branches:
MindMap
{%- else %}
Output: {%- endif %}
{%- endif %}
```

Figure 12: Iterative prompt in Jinja template format for mind map generation that is used in Algorithm 1.

```
{% set step = '{"node": "Global Climate change","branches
      ↪ ":[{"node": "effects","branches": [{"node": "
      ↪ melting ice"},{"node": "heat waves","branches": [{"
      ↪ node" : "droughts"}]}]},{"node": "causes","branches
      ↪ ": [{"node": "Enhanced greenhouse effect"},{"node":
      ↪  "Pollution","branches": [{"node": "Carbon emission
      ↪ "},{"node": "Burning coal"}]}]},{"node": "solutions
      ↪ ","branches": [{"node": "Individual efforts"},{"
      ↪ node": "International resolutions"}]}]}' -%}
A mind map is a diagram used to visually organize
      ↪ information into a hierarchy, showing relationships
      ↪  among pieces of the whole. It is often created
      ↪ around a single concept. Major ideas are connected
      ↪ directly to the central concept, and other ideas
      ↪ branch out from those major ideas. Mind maps can be
      ↪  generated based on the content present in text in
      ↪ multiple steps.
Consider the following example.
Given the following text:
Global climate change has many effects, including melting
      ↪ ice, heat waves, and droughts. It is caused by the
      ↪ enhanced greenhouse effect, which is caused by
      ↪ pollution, such as carbon emissions and burning
      ↪ coal. Solutions to global climate change include
      ↪ individual efforts and international resolutions.

Thought: Primary concept is Global climate change. Global
      ↪ climate change has branches, effects, causes and
      ↪ solutions. Effects have branches that include
      ↪ effects, melting ice and heat waves. Causes have
      ↪ branches enhanced greenhouse effect. Solutions have
      ↪  branches, individual efforts and international
      ↪ resolutions.

Output:
MindMap
{{ format_json(step)|safe }}

Now summarize the following text as a mind map.
{{input_text}}
```

Figure 13: Prompt in Jinja template format for mind map generation without iterative process.

```
Your task is to divide a passage into smaller passages
      ↪ grouped by similar facts. Separate passages by
      ↪ __NEW_PASSAGE__.
For example giving the following passage:
On December 31, 2016, the bank met all capital adequacy
      ↪ requirements to which it was subject and exceeded
      ↪ the regulatory minimum capital levels to be
      ↪ considered well-capitalized under the regulatory
      ↪ framework for prompt corrective action. At December
      ↪  31, 2016, the bank's ratio of common equity tier 1
      ↪  capital to risk-weighted assets was 11.59%, total
      ↪ capital to risk-weighted assets was 12.85%, tier 1
      ↪ capital to risk weighted assets was 11.59% and tier
      ↪  1 capital to average assets was 10.10%. Our
      ↪ shareholders are entitled to dividends when and if
      ↪ declared by our board of directors out of funds
      ↪ legally available. Connecticut law prohibits us
      ↪ from paying cash dividends except from our net
      ↪ profits, which are defined by state statutes. On
      ↪ January 27, 2016 the company's board of directors
      ↪ declared a $0.05 per share cash dividend, payable
      ↪ February 22, 2016 to shareholders of record on
      ↪ February 12, 2016. On April 27, 2016 the company's
      ↪ board of directors declared a $0.05 per share cash
      ↪ dividend, payable May 26, 2016 to shareholders of
      ↪ record on May 16, 2016. On July 27, 2016 the
      ↪ company's board of directors declared a $0.05 per
      ↪ share cash dividend, payable August 26, 2016 to
      ↪ shareholders of record on August 16, 2016. The
      ↪ company's board of directors declared a $0.07 per
      ↪ share cash dividend, payable November 28, 2016 to
      ↪ shareholders of record on November 18, 2016,
      ↪ representing a 40% increase when compared to the
      ↪ last quarter.
Smaller passages looks like:
__NEW_PASSAGE__
On December 31, 2016, the bank met all capital adequacy
      ↪ requirements to which it was subject and exceeded
      ↪ the regulatory minimum capital levels to be
      ↪ considered well-capitalized under the regulatory
      ↪ framework for prompt corrective action. At December
      ↪  31, 2016, the bank's ratio of common equity tier 1
      ↪  capital to risk-weighted assets was 11.59%, total
      ↪ capital to risk-weighted assets was 12.85%, tier 1
      ↪ capital to risk weighted assets was 11.59% and tier
      ↪  1 capital to average assets was 10.10%.
__NEW_PASSAGE__
Our shareholders are entitled to dividends when and if
      ↪ declared by our board of directors out of funds
      ↪ legally available. Connecticut law prohibits us
      ↪ from paying cash dividends except from our net
      ↪ profits, which are defined by state statutes. On
      ↪ January 27, 2016 the company's board of directors
      ↪ declared a $0.05 per share cash dividend, payable
      ↪ February 22, 2016 to shareholders of record on
      ↪ February 12, 2016. On April 27, 2016 the company's
      ↪ board of directors declared a $0.05 per share cash
      ↪ dividend, payable May 26, 2016 to shareholders of
      ↪ record on May 16, 2016. On July 27, 2016 the
      ↪ company's board of directors declared a $0.05 per
      ↪ share cash dividend, payable August 26, 2016 to
      ↪ shareholders of record on August 16, 2016. The
      ↪ company's board of directors declared a $0.07 per
      ↪ share cash dividend, payable November 28, 2016 to
      ↪ shareholders of record on November 18, 2016,
      ↪ representing a 40% increase when compared to the
      ↪ last quarter.

Now divide the following passage into Smaller passages
      ↪ grouped by similar facts.
{{input_text}}
```

```
Summarize the contents of the text below in a table.

{{input_text}}

Use the following format.

Caption: A caption for the table you generate. Can be
      ↪ multiple lines
Table: A table in markdown format.

Caption:
```

Figure 14: Text segmentation prompt (top) for multiple table generation. Zero-shot prompt for text to table and caption generation (bottom).

```
Given the text:
{{bullet_points}}

Table:
{{table}}

Rewrite the table adding a citation column, using the format
      ↪  [X], indicating the sentence number where that
      ↪ specific information can be found. When unsure use
      ↪ [NA].

Table with citations:
```

```
Input text:
{{bullet_points}}

Paths:
{{paths}}

Add an attribution to each path, using the format [X], where
      ↪  X is a sentence of the input text. When unsure use
      ↪  [NA] as attribution.

Paths with attribution:
```

Figure 15: Factuality critic prompts for Table (top) and Mind maps (bottom).

```
Your task is to check if all the values in a list falls
      ↪ under a category. Go over all the values one by one
      ↪  and check if they belong to the assigned category.
      ↪  Use the following format to answer.

Thought: Reasoning for the answer.
Answer: Single final answer yes or no.

Category: {{category}}
Values:
{{values}}

Thought:
```

```
There are some words or sentences that describes concept
      ↪ while other describes values associated with them.
      ↪ Values are defined as ordinals, type of job, degree
      ↪ , education level, location, region, date etc.
Answer No If any words is not a specific value otherwise
      ↪ answer yes.
For example:
Words:
Delhi
10
Cat
26 May
Lawyer

Thought: All the words are specific content words.
Answer: yes

Words:
IBM
Trucks
Birth
Family
26 May

Thought: Many words such as Trucks, Family, Birth are
      ↪ concept without specific values.
Answer: no


Words:
{{words}}

Thought:
```

Figure 16: Local structure critic prompt for Tables (top, zero-shot) and Mind maps (bottom, few-shot).

```
A TOC contains titles that are either too specific/long or
      ↪ too generic/short, or common sense. A useful TOC
      ↪ contains short, concise and informative titles at
      ↪ the right level of abstraction. Following a path in
      ↪  the TOC should allow you to generate or infer a
      ↪ sensible sentence.

Table of contents: root((K-26))
    1. southern terminus
      1.1. location
        1.1.2. city
        1.1.3. state
      1.2. state
    2. northern terminus
      2.1. location
        2.1.1. city
        2.1.2. state
      2.2. state
    3. maintained by
      3.1. organization
    4. traffic
      4.1. annual average daily traffic
      4.2. trucks
    5. national highway system
      5.1. listed
Thought (be specific): I can't create a sensible sentence
      ↪ following the path K-26 -> southern terminus ->
      ↪ location -> city
Useful: no
Table of contents: root((Assyria))
    1. Assyrian cities
      1.1. Aššur
      1.2. Nineveh
    2. Assyrian empire
      2.1. Neo-Assyrian Empire
    3. Assyrian period
    4. Assyrian kingdoms
      4.1. Adiabene
      4.2. Osroene
      4.3. Assur
      4.4. Beth Garmai
    5. Assyrian language
      5.1. Old Aramaic language
      5.2. Syriac language
Thought (be specific): All the titles contain useful
      ↪ information. All the paths allow generation of
      ↪ sensible sentences.
Useful: yes
Table of contents: root((Lonnie Johnson))
    1. Early life
      1.1. Birth
      1.2. Family
      1.3. Education
    2. Career
      2.1. Blues contest
      2.2. Recording contract
      2.3. Recordings
      2.4. Tours
      2.5. Collaborations
      2.6. Style
      2.7. Compositions
      2.8. Great Depression
      2.9. Later years
    3. Death
      3.1. Date
      3.2. Place
      3.3. Cause
Thought (be specific): Lonnie Johnson -> Early life -> Birth
      ↪  is a generic path not useful for generating a
      ↪ sentence.
Useful: no
Table of contents: {{toc}}
Thought (be specific):
```

Figure 17: Global structure critic few shot prompt for Mind map.

```
Your task is to generate a list of fact based questions that
      ↪  can be answered by the text passage. The format
      ↪ should be [Question][Answer].
Paragraph: {{text}}
```

```
Check if the following two answers are equivalent.
Use the following format.
Question: question text
Answer 1: answer text
Answer 2: answer text
Conclusion: Yes/No

Question: {{context_question}}
Answer 1: {{answer1}}
Answer 2: {{answer2}}
Conclusion:
```

```
Answer in concise manner the question using the information
      ↪ below. Say <unknown> when the questions cannot be
      ↪ answered.

{{data}}

Question:
{{question}}
```

Figure 18: Prompts used by the AutoQA pipeline: QA pair generation prompt *(top)*; Conditional answer equivalence *(middle)*; Question answering prompt *(bottom)*